In the Claims:

Please cancel claims 26 and 27, without prejudice.

Please amend claims 1, 3, 6, 14, 25 and 28 as follows:

1. (Currently amended) A processor comprising:

an instruction issuing unit which issues, in a predetermined sequence, instructions to be executed, said sequence of instructions including preselected value-producing instructions which, when executed, produce respective values;

at least one instruction executing unit which executes the issued instructions;

a register unit, having a plurality of registers, which stores values produced by the executed instructions;

a sequence number assigning unit which assigns said values produced by said value-producing instructions respective sequence numbers according to the order of issuance of their respective value-producing instructions; and

a register allocating unit which allocates each said produced value to one of said registers, for storing that produced value, in dependence upon the sequence number assigned to that value.

2. (Original) A processor as claimed in claim 1, wherein said register allocating unit is operable to allocate each said produced value its said register independently

of information contained in the value-producing instruction which when executed produces that value.

3. (Currently amended) A processor as claimed in claim 1, wherein said sequence of instructions also includes at least one preselected value-requiring instruction which, when executed, requires said produced value of a previously-issued one of said value-producing instructions, the processor further comprising an allocated register identifying unit operable, during execution of such a value-requiring instruction, to employ information contained in the value-requiring ~~instruction,~~ instruction to identify the register allocated for storing said produced value of said previously-issued instruction, said information being dependent upon said sequence number assigned to ~~said~~ that produced value ~~of said previously-issued instruction, to identify the register allocated for storing that value~~.

4. (Original) A processor as claimed in claim 3, wherein said information is a sequence offset representing a difference between the latest-assigned sequence number at the point, in said predetermined sequence, of issuance of said value-requiring instruction and said sequence number assigned to the produced value of said previously-issued instruction.

5. (Original) A processor as claimed in claim 1, wherein said register unit includes:

a set of physical registers allocatable for storing said produced values;

a mapping unit which maps logical register identifiers specified by said at least one instruction executing unit to respective corresponding physical registers of said set; and

a register renaming unit which changes said mapping between said logical register identifiers and said corresponding physical registers dynamically during operation of the processor.

6. (Currently amended) A processor as claimed in claim 5, wherein said register allocating unit is operable to allocate said produced value of each said value-producing instruction such that one of said physical registers which, in said mapping applicable at the point of issuance of that value-producing instruction, has a predetermined logical register identifier.

7. (Original) A processor as claimed in claim 5, wherein said register renaming unit is operable to change said mapping each time such a value-producing instruction is issued.

8. (Original) A processor as claimed in claim 7, wherein said register renaming unit is switchable selectively between a disabled mode, in which said mapping is not changed when such a value-producing instruction is issued, and an enabled mode in which said mapping is changed each time such a value-producing instruction is issued.

9. (Original) A processor as claimed in claim 7, wherein said register renaming unit is switchable selectively between a first renaming mode, in which said mapping is changed each time such a value-producing instruction is issued, and a second renaming mode different from said first renaming mode.

10. (Original) A processor as claimed in claim 9, wherein in said second renaming mode said mapping is changed each time a software pipeline boundary is crossed during execution of a software-pipelined loop.

11. (Original) A processor as claimed in claim 8, further comprising a mode register having one or more mode bits for specifying which of said modes said register renaming unit has.

12. (Original) A processor as claimed in claim 5, wherein:

the physical registers of said set are arranged one after the next at consecutive addresses in a renameable region of a register file; and

said mapping unit is operable to map a specified logical register identifier to its corresponding physical register using a mapping offset which represents a variable difference between the specified logical register identifier and said address, in said renameable region, of the corresponding physical register.

13. (Original) A processor as claimed in claim 12, wherein said register renaming unit is operable to change said mapping by incrementing or decrementing said mapping offset.

14. (Currently amended) A processor as claimed in claim 12, wherein:

said ~~information~~ specified logical register identifier is a sequence offset representing a difference between the latest-assigned sequence number at the point, in said predetermined sequence, of issuance of said value-requiring instruction and said sequence number assigned to the produced value of said previously-issued instruction~~; and~~

~~said logical register identifier is provided directly by said sequence offset~~.

15. (Original) A processor as claimed in claim 1, wherein said instruction issuing unit has a plurality of instruction issue slots, and is operable to issue a plurality of instructions simultaneously at different respective ones of the instruction issue slots; and

the processor has a plurality of instruction executing units corresponding respectively to said instruction issue slots, each operable to execute the instructions issued at its said corresponding instruction issue slot.

16. (Original) A processor as claimed in claim 15, wherein said sequence number assigning unit is operable, when two or more value-producing instructions are issued simultaneously at different respective instruction issue slots, to assign different respective such sequence numbers to the produced values of those two or more value-producing instructions according to a predetermined issue-slot order assigned to the respective instruction issue slots from which those instructions are issued.

17. (Original) A processor as claimed in claim 1, operable to execute the instructions of said sequence in a software-pipelined manner, wherein said preselected value-producing instructions include instructions which when executed will produce loop-variant values.

18. (Original) A processor as claimed in claim 1, further comprising a loop handling unit operable, in the event that a software-pipelined loop is found during execution of said instructions to require a zero number of iterations, to cause said sequence number assigning unit to skip one or more said sequence numbers before issuance of a first instruction following the loop.

19. (Original) A processor as claimed in claim 18, wherein the number of skipped sequence numbers is dependent upon the number of said value-producing instructions issued per iteration of said loop and a number of software-pipelined stages in said loop.

20. (Original) A processor as claimed in claim 18, wherein said loop handling unit is operable in said event to cause the instructions of said loop to be issued a number of times dependent on the number of software- pipeline stages in said loop whilst inhibiting said at least one instruction execution unit from executing those instructions, whereby each said value-producing instruction within said loop is issued said number of times.

21. (Original) A compilation method, for converting a sequence of high-level program instructions into a corresponding sequence of low-level instructions to be executed by a processor, the method comprising the steps of:

determining which said low-level instructions of said corresponding sequence are preselected value-producing instructions and which are preselected value-requiring instructions, each said value-producing instruction being an instruction which when executed will produce a value, and each said value-requiring instruction being an instruction which when executed will require said value produced by a previously-issued value-producing instruction;

assigning said produced values respective sequence numbers according to the order in which their respective value-producing instructions will be issued during execution; and

coding each said value-requiring instruction with information for use by said processor during execution to identify said produced value required by that instruction, that information being dependent on said sequence number assigned to that produced value.

22. (Original) A method as claimed in claim 21, wherein in said coding step said information is a sequence offset representing a difference between the sequence number assigned to the latest said produced value at the point in said corresponding sequence

at which said value- requiring instruction is issued and said sequence number assigned to said produced value required by that instruction.

23. (Original) A method as claimed in claim 21, wherein in said coding step each said value-producing instruction is coded without any information for use by the processor to identify where to store said produced value.

24. (Original) A method as claimed in claim 21, wherein said sequence of high-level program instructions includes a loop structure, the method further comprising the steps of:

analysing said loop structure to convert the high- level program instructions of the loop structure into a schedule of said low-level instructions to be executed iteratively by the processor according to a software pipeline; and

when one of said instructions in said schedule is such a value-producing instruction whose said produced value is a loop-variant value, assigning the produced value of that instruction different sequence numbers in different iterations.

25. (Currently amended) A computer-readable recording medium having stored thereon a computer program which, when run on a computer, causes the computer to carry out a compilation method for converting a sequence of high-level program

instructions into a corresponding sequence of low-level instructions to be executed by a processor, the computer program comprising:

a determining portion that determines which said low-level instructions of said corresponding sequence are preselected value-producing instructions and which are preselected value-requiring instructions, each said value-producing instruction being an instruction which when executed will produce a value and each said value ~~high~~-requiring instruction being an instruction which when executed will require the value produced by a previously-issued value-producing instruction;

an assigning portion which assigns the produced values respective sequence numbers according to the order in which their respective value-producing instructions will be issued during execution; and

a coding portion which codes each value-requiring instruction with information for use by the processor to identify said produced value required by that instruction, that information being dependent on the sequence number assigned to that produced value.

26-27. (Canceled)

28. (Currently amended) A processor comprising:

instruction issuing means for issuing, in a predetermined sequence, instructions to be executed, ~~the~~-said sequence of instructions including preselected value-producing

instructions which, when executed, produce respective values;

instruction executing means for executing the issued instructions;

register means, having a plurality of registers, for storing values produced by the executed instructions;

sequence number assigning means for assigning said values produced by said value-producing instructions respective sequence numbers according to the order of issuance of their respective value-producing instructions; and

register allocating means for allocating each said produced value to one of the said registers, for storing that produced value, in dependence upon the sequence number assigned to that value.